N92-22683

REAL-TIME, INTERACTIVE, VISUALLY UPDATED SIMULATOR SYSTEM FOR TELEPRESENCE

Frederick S. Schebor and Jerry L. Turney
KMS Inc.
Ann Arbor, Michigan 48106-1567

Neville I. Marzwell
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Time delays and limited sensory feedback of remote telerobotic systems tend to disorient teleoperators and dramatically decrease the operator's performance. To remove the effects of time delays, we have designed and developed key components of a prototype forward simulation subsystem, the Global-Local Environment Telerobotic Simulator (GLETS) that buffers the operator from the remote task. GLETS totally immerses an operator in a real-time, interactive, simulated, visually updated artificial environment of the remote telerobotic site. Using GLETS, the operator will, in effect, enter into a telerobotic virtual reality and can easily form a gestalt of the virtual "local site" that matches the operator's normal interactions with the remote site. In addition to use in space based telerobotics, GLETS, due to its extendable architecture, can also be used in other teleoperational environments such as toxic material handling, construction and undersea exploration.

INTRODUCTION

To reduce the number of dangerous extravehicular activities required of astronauts, NASA will need telerobotic and autonomous robotic systems [1], [2], [3]. These intelligent robotic systems will build structures in orbit, perform repetitive manufacturing tasks and conduct repair, resupply, and servicing. Since the costs, risks and the required logistical support for astronauts to accomplish these tasks are very high, the need exists to perform as many functions as possible autonomously under ground based supervisory control. Because autonomous robots are currently beyond the state-of- the-art, NASA must initially use telerobotic or shared telerobotic/autonomous systems employing human operators to direct remote robotic manipulators. A major problem with these systems, however, stem from the delays inherent in transmitting data (~6 sec) to and from the remote telerobotic site. Because of these delays, conventional teleoperation using instantaneous feedback is not possible. Instead, the teleoperator and the teleoperation must operate in a decoupled manner. Unfortunately, teleoperation with time delays causes enormous operator fatigue and, as a result, can cause a dramatic increase in the number of potentially dangerous operator errors [4]. To compensate for this, NASA's Jet Propulsion Lab (JPL) has developed concepts of shared control [5] and "phantom" telerobotic systems [4]. The latter provide real-time visual feedback to an operator by predicting the motion of the remote telerobot and by graphically simulating the telerobotic motion and presenting this data in real-time as a response to the operator's inputs. The problem with this approach is that simulations can never perfectly account for the physics of the remote site, and, as a consequence, the simulator will eventually fail to accurately predict the state of the remote site. Therefore, a mechanism is required to continuously monitor the state of the remote site and feedback differences to allow the simulator to be operated in a closed loop.

To simplify teleoperation, KMS has designed a Global-Local Environment Telerobotic Simulator, GLETS. The GLETS design (Fig. 1) eliminates the lag between operator input and visual feedback response by interposing a simulator between the operator and the remote site. GLETS:

- simulates a remote manipulator and its local environment,
- continually updates the simulated environment using remote visual sensors,
- manipulates the simulated environment with an easy-to-use, gesturally and voice controlled operator interface, which provides rich visual and audio cues that can be easily interpreted by the operator,
- uses CAD databases for the simulated robot environment and robot descriptions,
- provides a flexible object-oriented software architecture with underlying standard robotics algorithmic support, which results in software that is reusable, extensible, reliable, and portable.

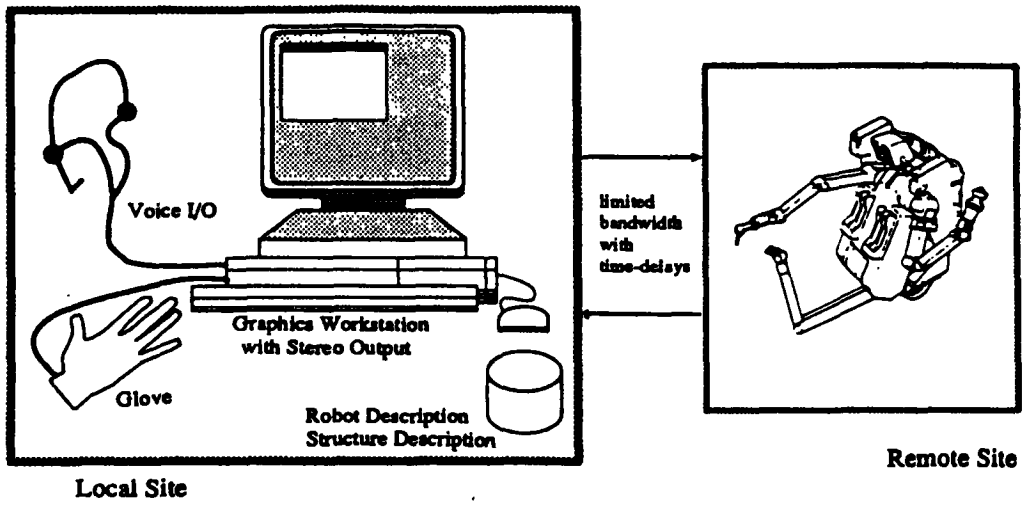

FIG. 1. By inserting GLETS between the operator and the remote manipulator, the simulator can shield the operator from problems of delays in the manipulator, interpreting the sensory input from the manipulators environment, and maintaining a detailed mental model of the manipulator's environment

OPERATOR INTERFACE

The effectiveness of a telerobotic system as a tool depends largely on the way the system is interfaced with the operator [6]. Due to the poor quality of telerobotic interfaces in the past, teleoperator training has been extremely expensive, and attempts have been made to identify favorable operator characteristics for shorter training time and safer system performance [7].

Current telerobotic simulators suffer from similar problems and many interfaces, such as those of ROBCAD and DENEB, concentrate on CAD tools for building up robot workcells. Because one must interact with the simulated robot through a 2-D graphics screen, it is difficult to properly control the robot manipulator arm in its simulated three-dimensional environment. In addition, the flow of information and commands between the simulator and the operator are restricted by the current interface technologies of CRT terminals, keyboards, joysticks, and mice [8]. Users are required to structure and channel communication to fit the machine. The key to solving this problem is to make the interface to the simulator more "human-like" rather than requiring the operator to be more "machine-like."

These problems can be eliminated, in a sense, by transporting the operator to the remote site so that the task can be accomplished and monitored handily. GLETS implements this concept by totally immersing an operator in a real-time, interactive, visually updated, simulated environment of the remote telerobotic site. Using GLETS the operator will, in effect, be able to interact with a telerobotic virtual reality and to form a gestalt of the virtual "local telerobotic site" that matches the operator's interactions with the actual remote site. These capabilities are provided in the following ways.

Display

Visually, GLETS portrays the telerobotic environment as a three-dimensional, shaded, color graphics world (Fig. 2). The simulator provides stereo images by using polarized glasses that alternately block the left and right lens synchronized with a graphics display that portrays the simulated environment from two slightly different views. Shading, perspective, motion, and stereo are the strongest depth cues in the human visual system. By incorporating these cues, GLETS takes maximum advantage of the operator's visual system, thereby making the task of controlling the arm more natural to the operator, and providing a more realistic simulated environment.

The graphics screen of GLETS can be divided into a number of windows. The windows display the views that would be seen by a set of "pseudo-cameras" (Fig. 3). Pseudo-cameras are attachable to any point within the simulated environment. For example, the operator can attach a pseudo- camera to the gripper of the manipulator to provide a continuous view of what would be seen by a gripper-held camera. Or, for a particularly difficult task such

FIG. 2. GLETS output is in the form of 3-D stereoscopic, shaded surface graphics and synthesized speech.

as gripping an object from behind, the operator could attach a pseudo-camera to the grip position on the object and use it to guide the gripper to the proper position.

GLETS allows the feeling of depth, commonly referred to as "stereo disparity," to be changed. Disparity is the difference in view angle registered between two eyes. Therefore, in general, human stereo is useful only for objects that are in close proximity. GLETS allows this disparity to be controlled in order to allow distant objects within the simulated environment to have the same perception of depth as closer objects.

GLETS provides multiple light sources (pseudo-lights) that can be positioned by the operator's hand and attached anywhere in space (Fig. 4). This permits elimination of shadows in critical areas.

GLETS allows any object in the simulated environment to be made transparent in order to let internal or difficult-to-see objects be viewed. For example, if the operator is required to insert a bolt into a particular hole and it was

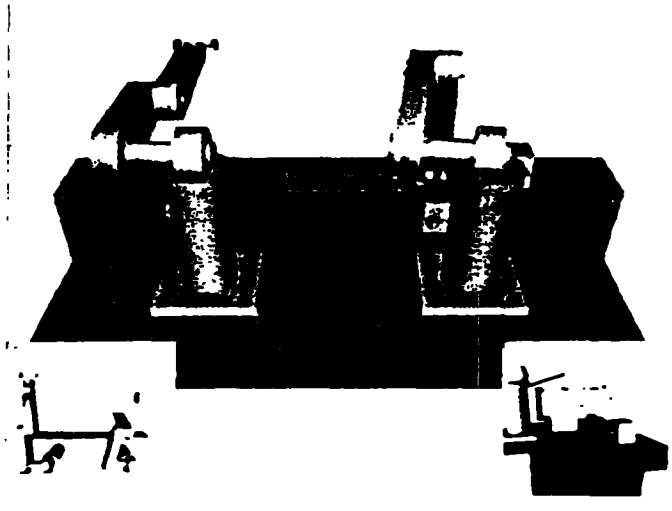

FIG. 3. The GLETS interface permits the operator to define and display multiple windows into the remote site

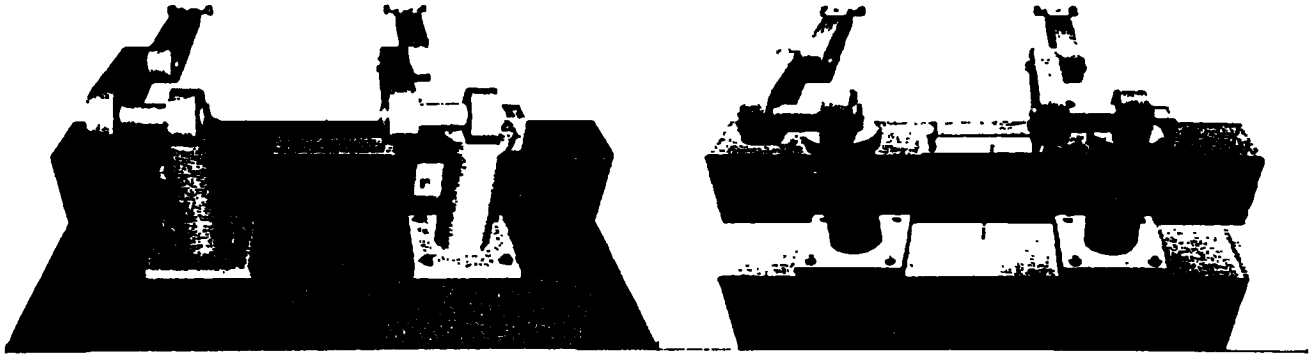

FIG. 4. The GLETS interface allows the operator to define multiple light sources (at infinity or local) at arbitrary locations.

not clear to the operator where the hole is located or how the hole is oriented, the entire object that contains the hole could be made transparent except for the inside surface of the hole in order to show the operator its position (Fig. 5).

Another visual attribute associated with objects in the simulated environment is the "glow" attribute which may be used to show the operator grip locations on an object or other special locations. These points are identified with a special color.

To test new control algorithms, GLETS provides on-screen information on the joint positions, velocities, accelerations, forces and torques (Fig. 6).

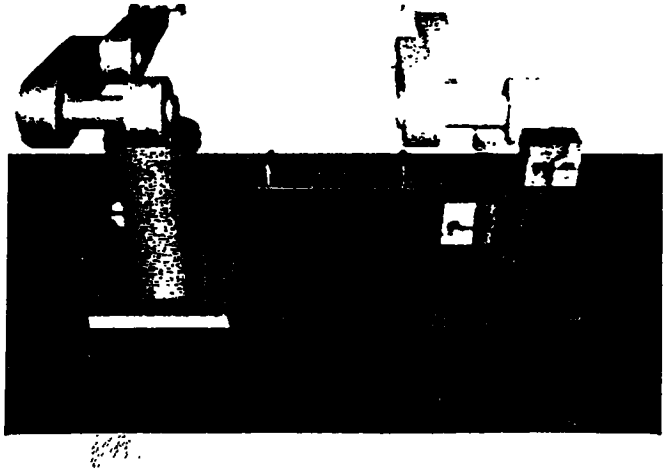

FIG. 5. With GLETS the operator can turn any objects at the remote site transparent in order to obtain a better relational understanding of the objects in the environment.

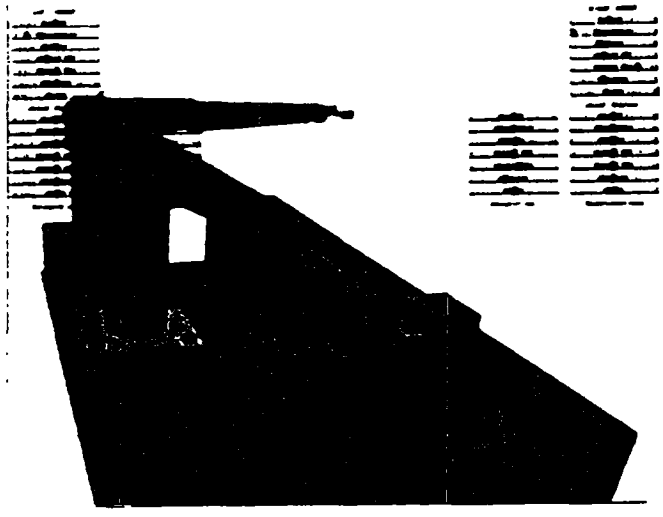

FIG. 6. The GLETS interface permits the operator to graphically view the physical parameters of robot motion.

Finally, in addition to shaded surfaces, GLETS can display any or all objects as wireframes (Fig. 7).

Computer-Linked Glove

A computer-linked glove [9] that allows the system to monitor the position, orientation and finger configuration of an operator's hand, permits the operator to use his hand to control the end effector. As the operator moves his hand, the system monitors the changes in position and orientation and passes these through reverse kinematics equations to determine the changes in the manipulator's joint angles necessary to match the hand position. The glove also permits the hand to be used as a "pick" device on virtual control panels, or through the use of gestures, to provide complex instructions to the telerobotic simulator, such as grasp, release, insert, extract, open, or close.

These complex instructions are implemented as a set of macros, or subprograms, that have been developed for

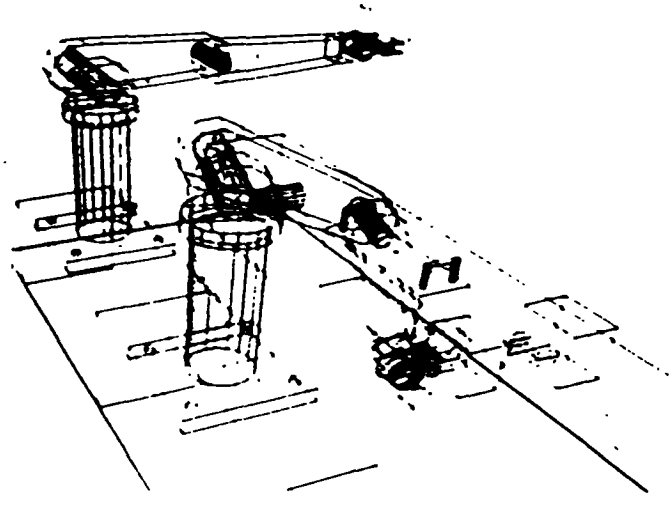

FIG. 7. The GLETS interface permits the operator to display objects as wireframes with hidden-line removal or with labeled object axis.

special applications. Macros are used for two reasons. First, to simplify an operator's interaction with the environment. For example, an operator only need position a telemanipulator in the approximate vicinity of a handle, and execute the "grasp" macro either through a verbal command (described in the next sub-section) or through a gesture and the preprogrammed macro will "take over" and go through the necessary steps to grasp the handle. More importantly, however, macros provide a form of shared control [5] in which the telerobot can be controlled during free movement in the teleoperated mode, but must be controlled in autonomous mode during contact operations.

In addition to manipulating simulated objects, non-physical tasks, such as changing control parameters on a manipulator arm, are performed with object-like metaphors. To change a parameter, for example, the operator is provided with simulated slide bars and knobs that can be hand positioned using the glove.

Speech Synthesis and Voice Recognition

To provide input of more abstract commands than can be input through gesturing, and to notify the operator of simulation status, the GLETS system employs a speech synthesizer and a voice recognition unit. Voice input and output have produced productive gains in graphical workstation environments [10], [11], [12]. In GLETS, verbal commands are used to set up system states, execute macros and to receive status. This type of interface promotes easy, reliable control of the GLETS system. With little training an operator can quickly master the entire control of the system.

VISUAL UPDATE

Unless a robot simulator can maintain both a correct and current world model of its environment, it cannot function in a telerobotic application. To address this need, KMS has designed a software architecture and visual update subsystems that allow the simulator to interact with a remote vision system for update of the world model.

The update capability, in a sense, closes the teleoperational control loop by providing sensory feedback that can be used to minimize the discrepancies between the simulated and actual telerobotic environment. Without feedback, an open loop simulation can not completely characterize the remote environment. Therefore, discrepancies would start to appear that would be difficult for an operator to cope with. In particular, contact forces are notoriously difficult to model. If these are unknown, as is generally the case, then, an object could potentially slip in a gripper at the remote site and this slippage would not be incorporated in the telerobotic simulation. As a consequence the object might later collide with other objects in the environment, or the object might not be positioned correctly by the manipulator when it was used in a later insertion operation.

The visual update system serves two purposes:

1. to determine the minor discrepancies between the real and simulated world environments, and to update the simulator when these discrepancies become too large to tolerate,
2. to locate objects that are completely lost by the simulator.

Methods

To perform the first function, the visual update system makes real-time range (distance) data and image data measurements of the telerobotic environment. This could be done, for example, using a full-field image/range camera (such as described in the following sub-section). The range and image data can then be compared to the simulated range and image data generated by GLETS. JPL has used a similar approach [4] with single image video data.

In the same way that image data can be compared to simulated image data, range data can be compared to simulated range data, generated from the z-buffer data (used for hidden surface calculations) of the simulator, to determine discrepancies. Once a discrepancy is discovered, the visual update system determines the correct pose of an object using KMS developed vision algorithms, and uses this pose to relocate the object in the simulator. In GLETS we use an approach similar to JPL [4] to calibrate real and simulated image/range data. Specifically, points in the real environment (imaged by the range camera) and corresponding points in the simulated environment are hand selected. These are used to establish a mapping between the two environments.

When real and simulated objects are "out of sync" or when objects are completely lost by the simulator, GLETS employs an algorithm [13] that uses range (3-D) and/or image (2-D) data to locate 3-D objects. We refer to this algorithm as the RISER (Recognition by Iterative Spring Energy Reduction) algorithm. This robustness of this algorithm permits it to operate in cluttered environment, i.e., does not require that objects be completely visible.

Range-based RISER

In the range based RISER algorithm, 3-D model data representing objects that may appear in a scene, are matched to the range image of the scene. RISER requires that both the model and range data are represented as a collection of overlapping surface patches. Descriptors are then created by pairing up surface patches which are chosen from different regions of the surface. Descriptors of this type have a number of geometrical attributes, such as the angles between the normals at the center of each patch, the principle curvatures at each patch, etc., that allow them to be differentiated from each other.

To obtain a match, RISER sets up a set of pseudo forces between similar model and range descriptors. In effect, this is like connecting a 6-dimensional spring (3 translational and 3 rotational dimensions) between the descriptors, with a spring constant proportional to the descriptor's similarity (Fig. 8). The spring attempts to pull the corresponding model and range descriptors into alignment. To obtain convergence to a correct solution, the algorithm employs a robust statistical approach of reweighting the spring constants.

Image-based RISER

The first step of the image-based RISER algorithm is to construct shape representations of all objects in the image. As in the range-based algorithm, shape representations consist of a set of spatially local shape primitives described by various geometric attributes. The algorithm then uses an initial estimate of the object's pose in conjunction with a 3-D model of the object to predict the appearance of the edge contours. The predicted edge contours' shapes are represented by shape primitives in the same manner as the observed edge contours, permitting the observed shape primitives to be directly compared to the predicted shape primitives. The algorithm then connects pseudo-springs between the detected primitives and the predicted primitives, and as in the range-based RISER algorithm, the model is then freed and allowed to relax to the equilibrium pose under the influence of the spring forces.

The RISER pose determination approach has a number of advantages over conventional approaches. First, since it compares only pairs consisting of one image/range shape primitive and one predicted shape primitive, it avoids searching the exponentially large space of all possible image feature to model feature correspondences typical of many conventional approaches. Second, since RISER is comparing 2-D predicted contours to 2-D image/range contours, no assumptions about the nature of the surfaces of the objects are necessary.

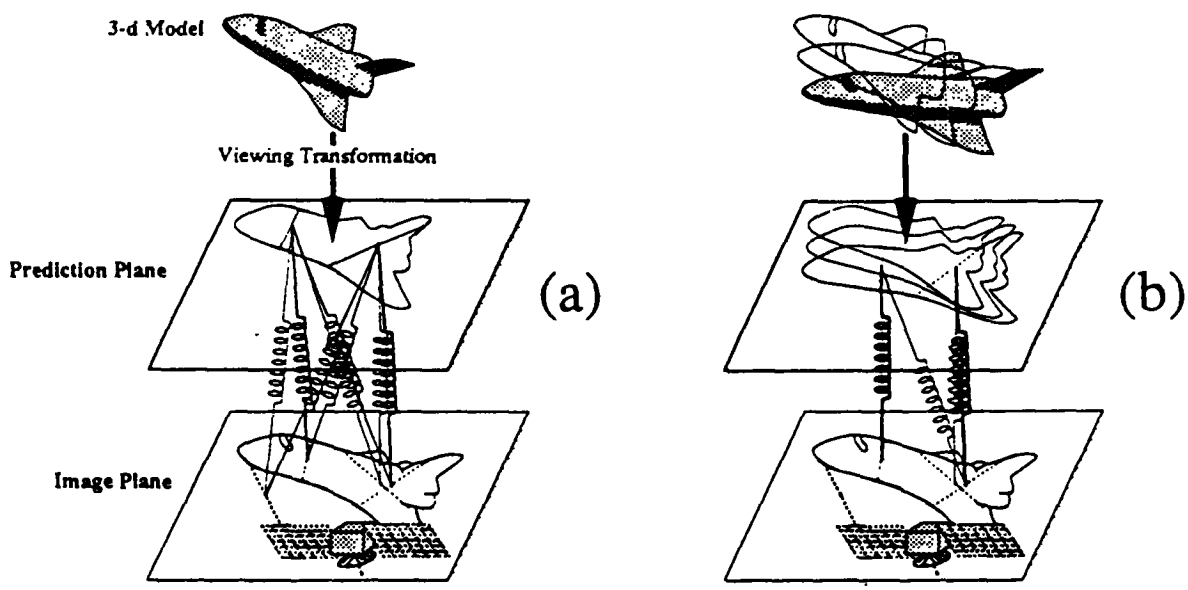

FIG. 8. RISER's advanced pose determination modules function, in effect, by connecting "springs" whose strengths are proportional to the similarity between features between primitives, a few of which are shown in (a). The model is then freed and allowed to relax into equilibrium, as in (b). The spring constants are then readjusted and the process continued until no further improvements occur.

Sensors

To address the need for visual tracking sensors, KMS has designed two different cameras that use structured lighting to extract range data. The RTRT camera, a Real-time Three-dimensional Range Tracking Sensor combines a rugged optical design with state-of-the-art phase-shift moiré interferometry [14]. The RTRT Camera generates moiré images, that encode the range of objects in the field of view, at an instant in time.

KMS is currently developing the other camera type for a "View-Generated Database" (VGD) project for NASA [15]. Our investigation into this problem has led KMS to a new type of structured lighting wherein phase shifted cosine patterns are projected using high-quality sinusoidal slides onto the surfaces of the scene using a CCD camera to image them. The phase values obtained by the CCD camera are used to triangulate the distance to a surface. We call this new technique SUrface Reconstruction by PHAse-shifted CosinEs (SURPHACE).

Using either of these image/range camera designs or other systems that are currently being developed, the GLETS visual update system is able to bring in image and range data that can be directly compared to calibrated, simulated data to determine where discrepancies have occurred.

REMOTE MANIPULATOR SIMULATION

The GLETS design also addresses the completeness of the simulation for the manipulator and its environment. Full simulation includes graphical, geometric, kinematic, dynamic, and control simulations [17]. Graphical simulation is concerned with simulating the appearance of the manipulator for human interpretation. Geometric simulation is concerned with determining the spatial relationships of the parts of the manipulator and, for example, determining if collisions have, or are about to, occur. Kinematic simulation is concerned with determining the joint positions, velocities, and accelerations necessary to yield desired manipulator trajectories. Dynamic simulation includes simulation of the torques and forces acting on the manipulator resulting from motions of the manipulator and any attached objects. Control simulation involves the determination of the proper torques, forces, and motor currents necessary to achieve a particular motion of the manipulator.

In the GLETS object-oriented environment (described in the next section) graphical, geometric, kinematic, dynamic, and control attributes are associated with the robot arm links and actuators. For example, if a new type of actuator motor were developed, the simulator could be quickly updated by developing a new subclass that would describe the object and would know how to model the dynamic and control behavior of the new motor.

To provide a graphics description of the arm, GLETS inputs an IGES formatted file of the manipulator from a Robot Descriptor File (RDF). Manipulator links then render themselves to provide a three-dimensional graphics model of the manipulator.

For the kinematic description of the arm, GLETS inputs data from the RDF, that contains the Denavit-Hartenburg kinematic parameters for the manipulator, the most common method of describing robot linkages. The ranges of the joint angles are also supplied, so that once the full kinematic and dynamic description of the robot is input, the robot is assembled and actuators will constrain themselves to move within the space of realizable joint positions. The dynamic description, also stored in the RDF, provides masses and inertias for all of the robot links as well as viscous friction coefficients for the joints. Using this data, GLETS is able to determine the forces and torques at any point on the manipulator.

KALI

The KALI subroutine library provides a flexible robot programming and control environment for coordinated multi-arm robots and has been used extensively by JPL [16]. Although KALI has been designed for coordinated motion of manipulators, the GLETS system will, a large fraction of the time, use the components of KALI that pertain to describing the kinematics and dynamics of individual manipulators.

For example, for a given trajectory (defined by the operator using the computer-linked glove in the operator interface) of the end effector of the manipulator, KALI routines are used to solve the manipulator arm equations to determine the joint angle trajectories. Although the operator's motion may be somewhat erratic, trajectories are approximated appropriately so that they consist of a sequence of straight line segments connected by smooth arcs. The acceleration is chosen to be zero when the manipulator is in the middle of a segment and a constant non-zero value during the transition between segments. Given the force and torque constraints of the manipulator, KALI subroutines are then used to limit the accelerations along the trajectory to values that will not violate these constraints.

SOFTWARE ARCHITECTURE

While the operator interface and the completeness of the simulation are critical to the success of the telerobotic simulator, also important is the simulator's flexibility and extendibility. An inflexible and unextendable simulator would quickly become obsolete as manipulator technology advances. Therefore maximum flexibility in the creation, updating, and customization of manipulator simulation models is important. Elements and parameters of existing models should be simple to change and update. The software architecture should include simple mechanisms for users to extend the simulator's software to handle wholly new manipulator designs as well as incorporating new kinematic, dynamic and control techniques.

One way of dealing with these problems in the GLETS design is to is use object-oriented programming [18], [19], [20] in describing the telerobotic environment. By using an object-oriented design GLETS has a software architecture based on the objects in the telerobotic environment rather than on the functions that are performed.

Functions tend to be more abstract and tied to a particular implementation. Objects, on the other hand, are more physical and provide a more meaningful metaphor for designing code. The GLETS system performs actions on certain objects within the defined environment, and can be used in a evolutionary implementation. The software used in simulating the telerobot can be viewed as an operational model of the world in which the robot exists. GLETS is organized around representations of the objects in the telerobotic environment so that the software structure reflects the physical structure of the telerobotic environment. Complexity of the GLETS system is controlled by creating, combining, and manipulating software objects instantiated from a set of generic software classes to perform the specific tasks of the system. For example, an orbital replacement unit, ORU, class could be defined as shown in Fig. 9.

CONCLUSION

The GLETS system provides an operator with sensors, operator interface, and software-architecture that uniformly treats a simulated telerobotic environment as being populated with a set of physical objects that can be viewed, manipulated, and coded in a highly intuitive fashion. This capability greatly simplifies training and use of the GLETS system.

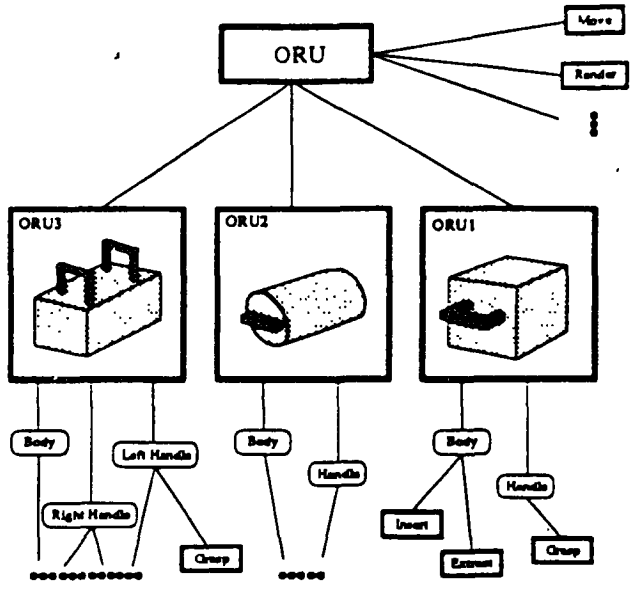

FIG. 9. GLETS object oriented software architecture supports the use of base and derived classes, a technique that greatly simplifies adding new objects to the system, because much of the properties of a new class of object can be inherited from its base class.

The most obvious use of GLETS will be in monitoring the state of a remote telerobot in a telerobotic operational loop in which severe time delays are expected. Although such conditions occur more severely between ground and space, they also occur in earth bound tasks such as undersea exploration, toxic waste management, remote inspection in nuclear and biological environments, and supervision in multiple automated production units. The GLETS system, because of its object oriented design, could be easily used in these areas as well as more generic applications such as evaluating new robotic algorithms as they evolve, facilitating robot configurations, mission planning/analysis, contingency planning, and error analysis.

ACKNOWLEDGMENTS

The authors wish to thank Arnold H.C. Chiu and Paul G. Gottschalk for their research efforts during this contract. The research described in this paper was partially carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work was funded by NASA Small Business Innovative Research Contract NAS7-1074.

---

[1] M. L. Reiss, "Automation and robotics and the development of the space station - U.S. Congressional view," *Advances in Astronautical Sciences*, 60, 531, 1986.

[2] NASA Task Force, "Robotics for the United States space station," *Robotics*, 1, 205, 1985.

[3] R. M. Hord, *Handbook of Space Technology: Status and Projections*, CRC Press, 1985.

[4] A. K. Bejczy, W. S. Kim, and S. C. Venema, "The phantom robot: Predictive displays for teleoperation with time delays," *IEEE Conf. on Robotics and Automation*, 546, 1990.

[5] S. Hayati and S. T. Venkataraman, "Design and implementation of a robot control system with traded and shared control capability," *IEEE Conf. on Robotics and Automation*, 1310, 1989.

[6] C. S. Hartley, D. J. Cwynar, K. D. Garcia, R. A. Schein, "Capture of satellites having rotational motion," *Proc. of 30th Annual Meeting of the Human Factors Society*, 875, 1986.

[7] J. P. Yorchak, C. S. Hartley, and E. Hinman, "Characterization of good teleoperators: What aptitudes, interests, and experience correlate with measures of teleoperator performance," *Proc. of 29th Annual Meeting of the Human Factors Soc.*, 1135, 1985.

[8] M. B. Friedman, "Gestural control of robot end effectors," *Proc. SPIE Intelligent Robots Computer Vision*, 726, 500, 1986.

[9] T. G. Zimmerson, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill, "A hand gesture interface device," *Proc. of the SIGCHI Conf of the ACM*, 189, 1987.

[10] G. L. Martin, "The utility of speech input in user-computer interfaces," *Int'l J. Man-Machine Studies*, 30, 255, 1989.

[11] C. Schmandt, M. S. Ackerman, and D. Hindus, "Augmenting a window system with speech input," *IEEE Computer*, 23, 50, 1990.

[12] M. W. Salisbury, J. H. Hendrickson, T. L. Lammers, C. Fu and S. A. Moody, "Talk and draw: Bundling speech and graphics," *IEEE Computer*, 23, 59, 1990.

[13] P. G. Gottschalk, *Machine Recognition and Attitude Estimation of 3D Objects in Intensity Images*, Univ. Michigan Ph.D. Thesis, 1990.

[14] G. T.Reid, R. C. Rixon, and H. I. Messer, "Absolute and comparative measurements of three-dimensional shape by phase measuring moiré topography," *Opt. and Laser Tech.*, 16, 315,1984.

[15] J. L. Turney, "High performance view-generated database for world model definition and update," KMSF-U1923, NASA Contract NAS7-1009, 1987.

[16] P. Backes, S. Hayati, V. Hayward, and K. Tso, "The KALI multi-arm robot programming and control environment," *Proc. of NASA Conf. on Space Telerobotics* , 1989.

[17] T. N. Mudge and J. L. Turney, "Unifying robot arm control," *IEEE Trans. on Ind. Appl.*, 1A-20, 6, 1554, 1984.

[18] B. Meyer, *Object-oriented Software Construction*, Prentice Hall, 1988.

[19] R. A. Volz and T. N. Mudge, "Robots are (nothing more than) abstract data types," *Robotic Research: The Next Five Years and Beyond* , 1984.

[20] D. J. Miller and R. C. Lennox, "An object-oriented environment for robot system architectures," *IEEE Conf. on Robotics and Automation*, 352, 1990.